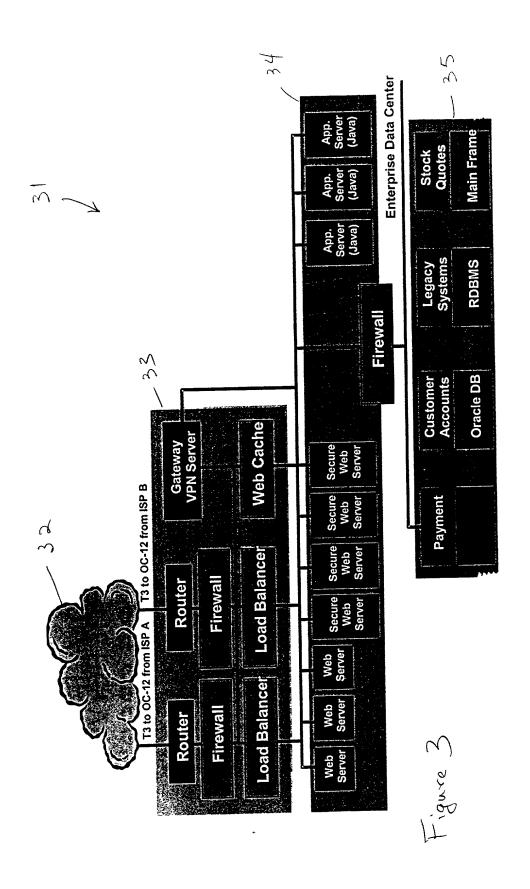


Figure 2. Data Encapsulation Through the Network Stack (Prior Art)



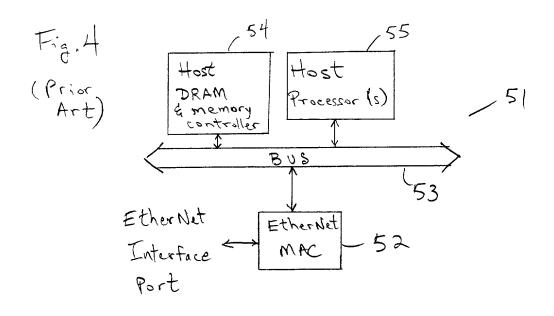
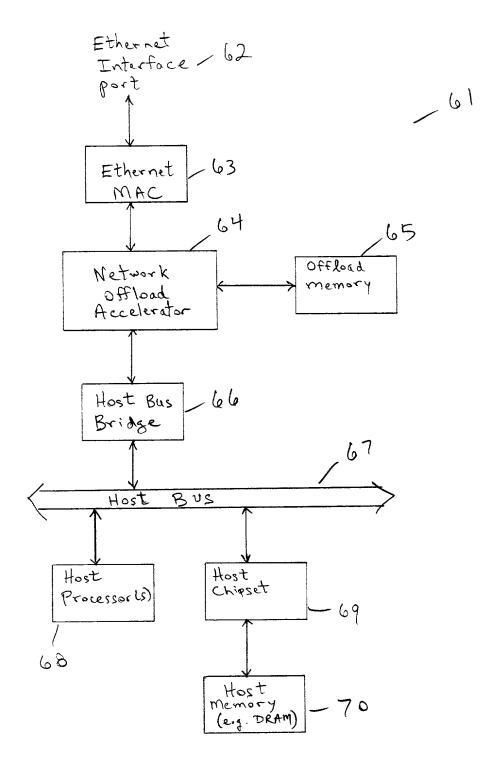
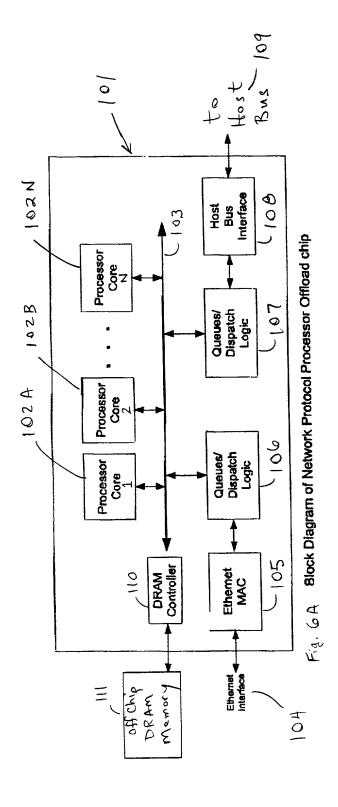
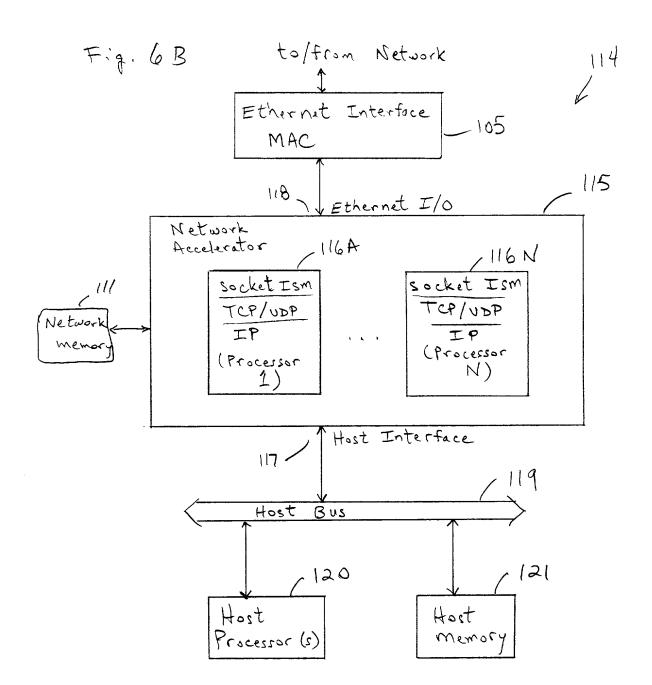
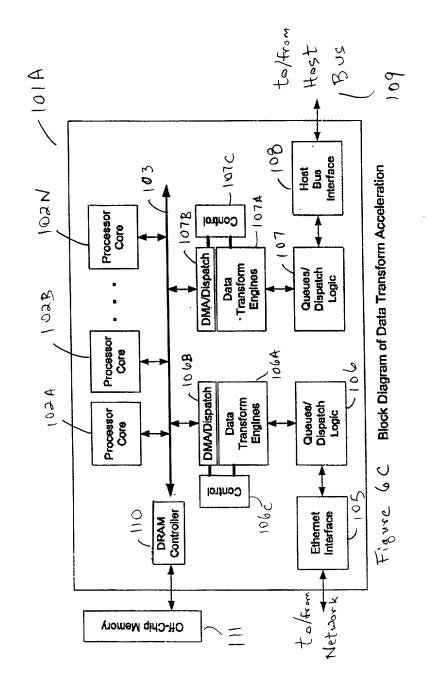


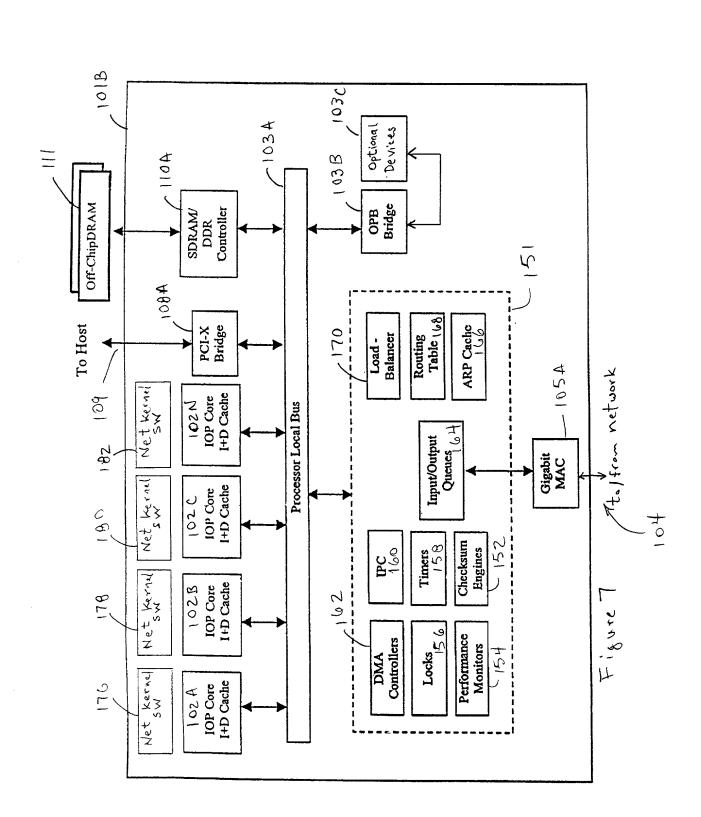
Figure 5 (prior art)











data and application data) at a single network interface port

203

205

207

209

distributing, from a single network interfoce port, a first group of network packets to a 1st processor which is executing a 1st network protocol stack in order to process the first group of network packets

distributing, from the single network interface port, a second group of network packets to a second processor which is executing a 2nd network protocol stack in order to process the second group of network packets (the 1st and 2nd network protocol stacks are separate threads)

after the 1st processor processes the 1st group by executing the 1st network protocol stack, 1st data associated with the 1st group is transmitted to a host bus interface and, through a DMA exerction, this 1st data is written to host memory

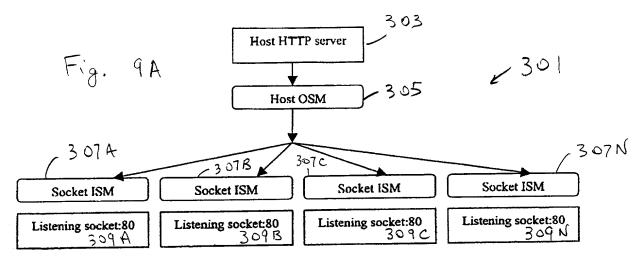
after the 2nd processor processes the 2nd group by executing the 2nd network protocol stack, 2nd data associated with the 2nd group is transmitted to the host bus interface, and, through a DMA operation, this 2nd data is written to the host memory

Fig. 83

receive 1st & 2nd application data (e.g. receive through DMA speration from 231 Host memory to network processing system viewory) determine tags associated with 1st + 2nd application data to determine which processes headers for both & out & 235 Prepare 1st packet header data associated with 1st application data in 1st processor which is executing 1st network protocol stack 237 Prepare 2nd packet header data associated with 2nd application data in 2nd processor which is executing 2nd network protocol Stack 239 Transmit 1st application data and 1st packet header data through a single network interface port Transmit 2nd application data and 2nd packet header data through the single network interface port

Fig. 8C 261 receive packet with application data and receive packet header data 263 examine field in packet header data (e.g. perform hashing operation on facket header data) 265 direct packet header data to one of 1st or 2nd processors based on output hashing operation 267 Store, through DMA operation(s), application data in 1st or 2nd portions of network processor memory which are preallocated respectively for 1st

and 2nd processors



Replication of listening sockets

Fig. 9B

312

Initialize OSM and ISM

(e.g. ISMs transmit number of network protocol processors (IOPs) and each ISM transmits on IOP-specific houdle to the OSM)

, 314

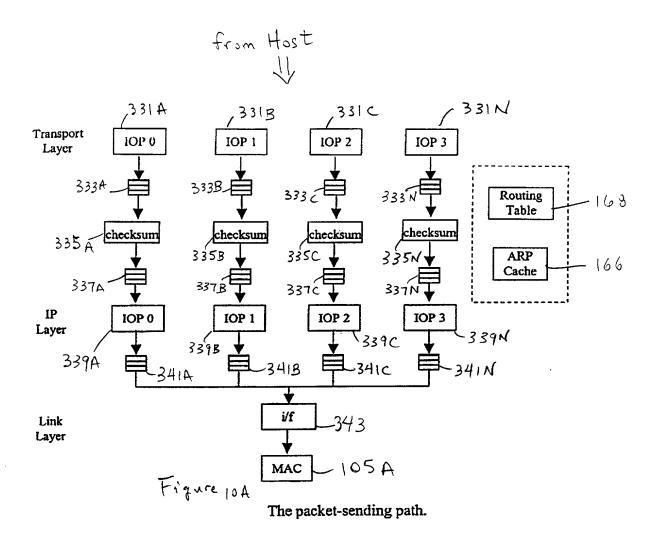
After initialization, a host application establishes a socket connection, using socket API calls

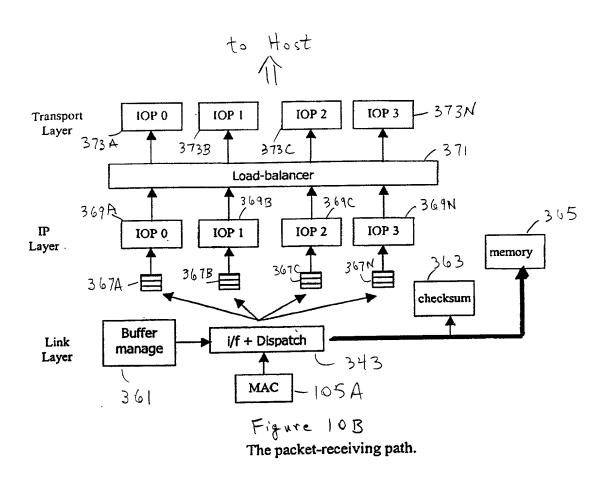
316

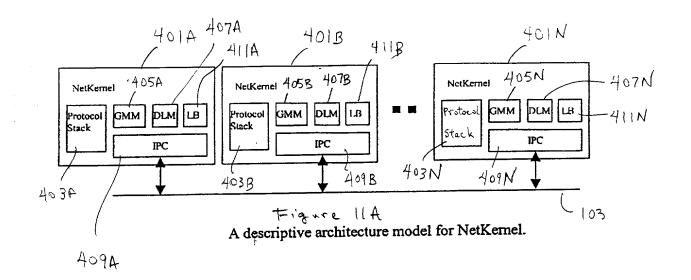
A client application connects to the socket the server application is listening on

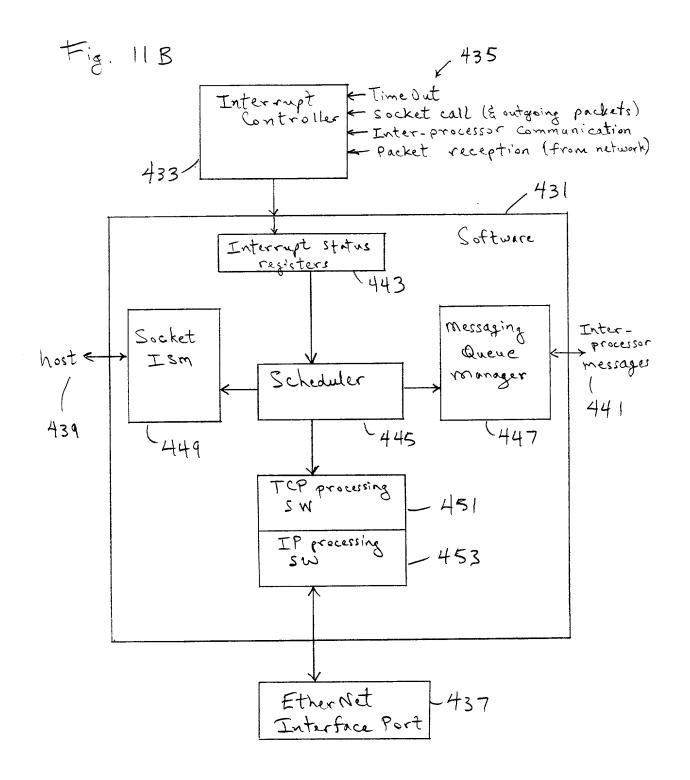
.318

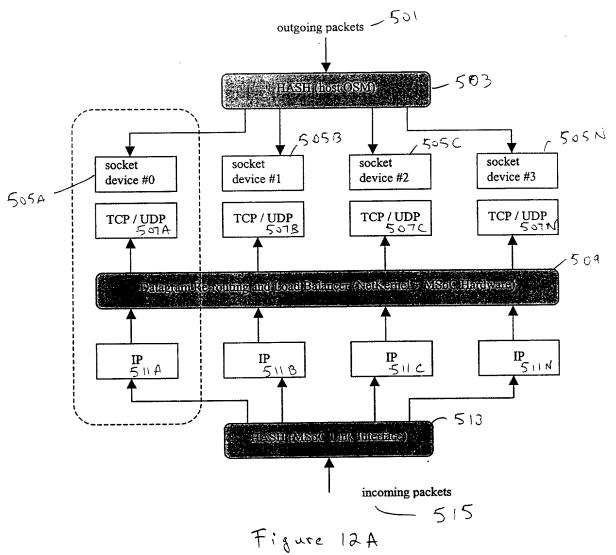
The OSM then receiver the client request, invoking the proper server application by identifying it from the OSM handle it allocated previously, and recording the associated ISM handle and IOP number in a table. The server application builds a response to the client. It transmits this request to the client, via the proper IOP, by prefixing the response using an form handle, ISM handle & message directed to the proper IOP by posting it to the IOP's IQP.



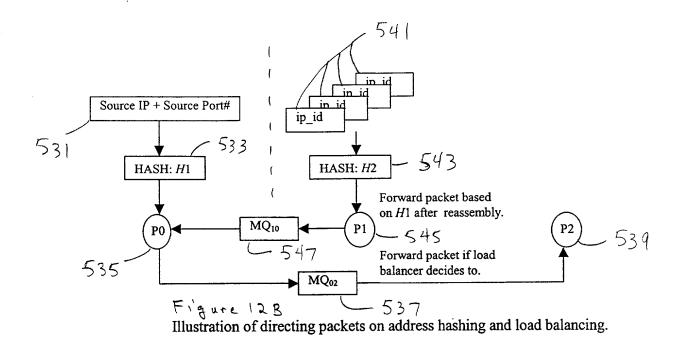








An illustration of packet flows for load balancing.



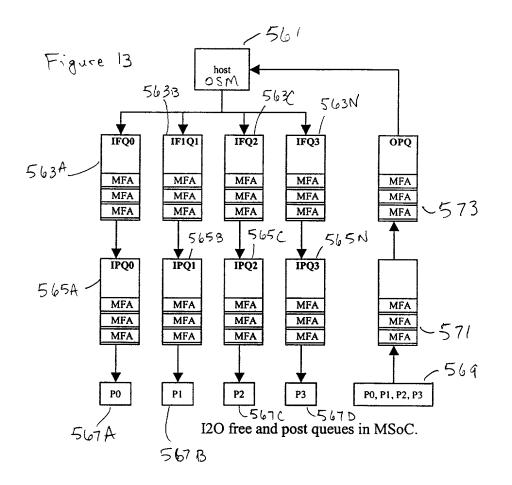


Figure 14

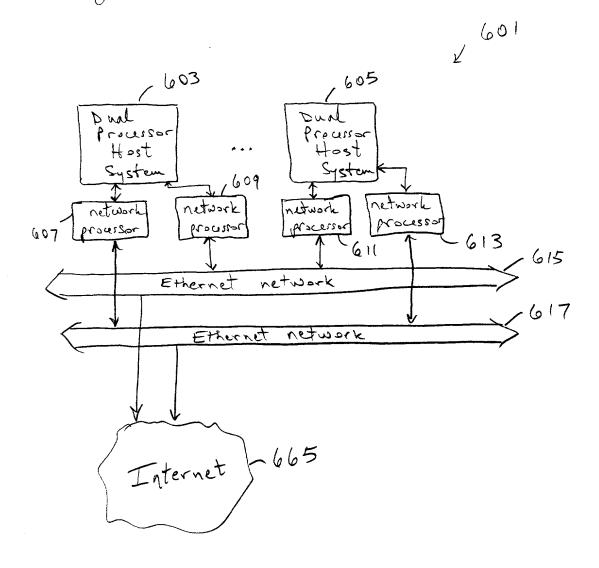
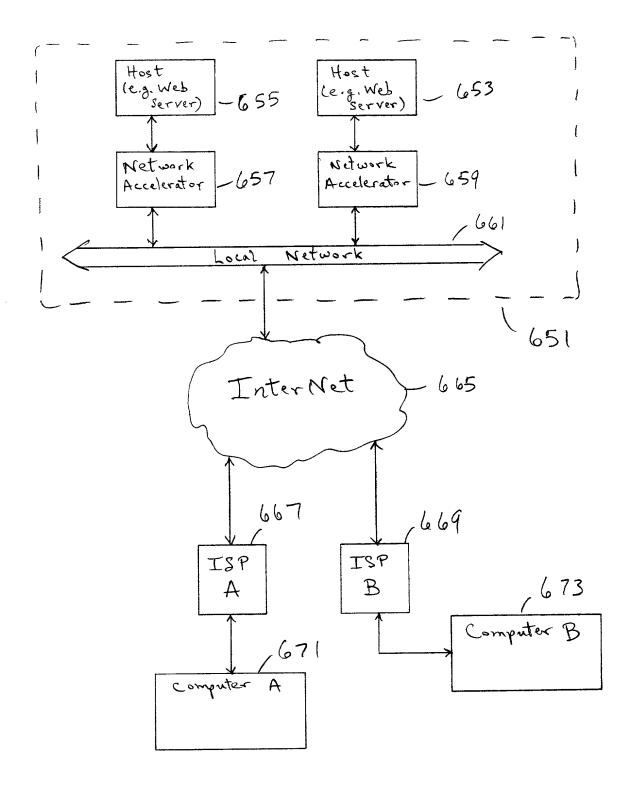
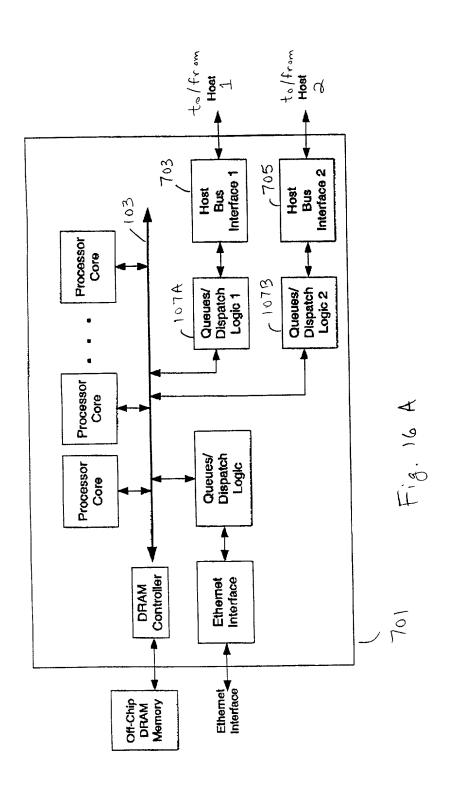
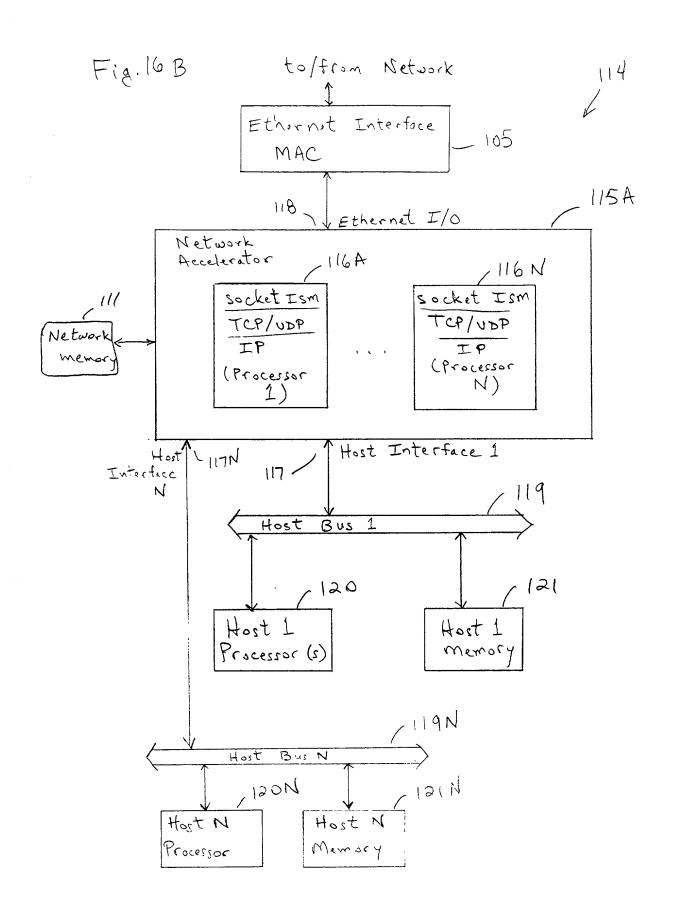


Fig. 15







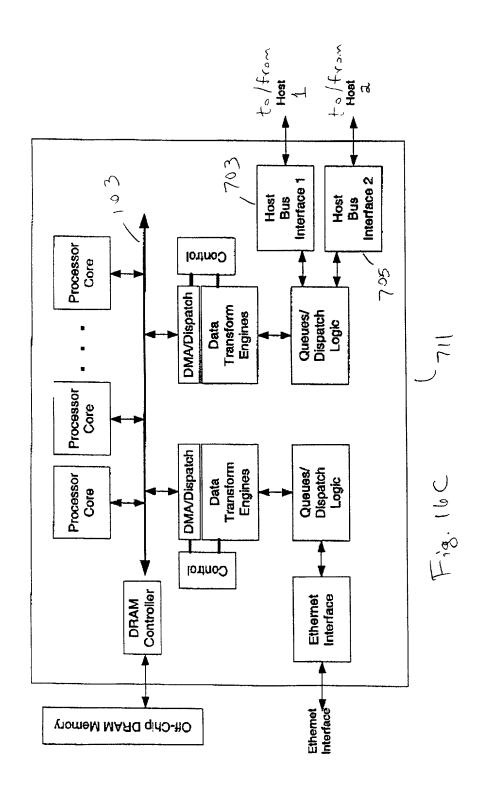
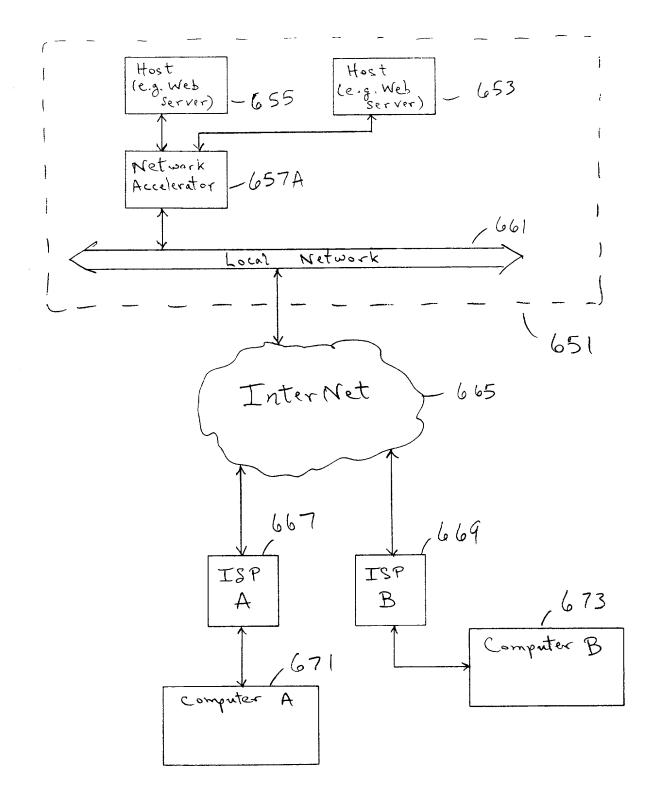


Fig. 16D



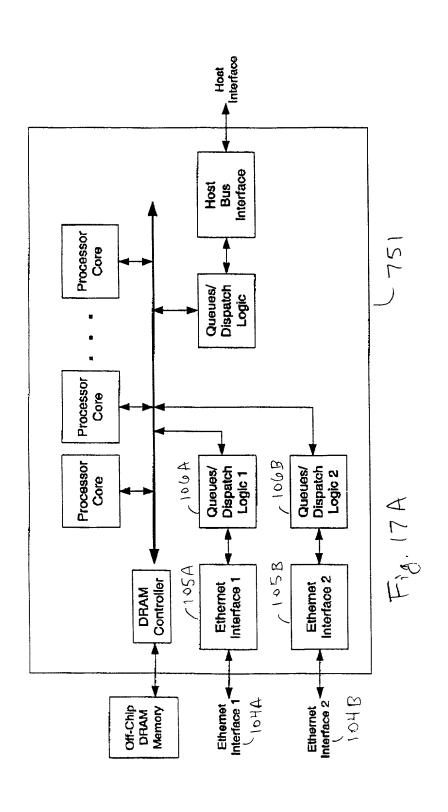


Figure 17 B

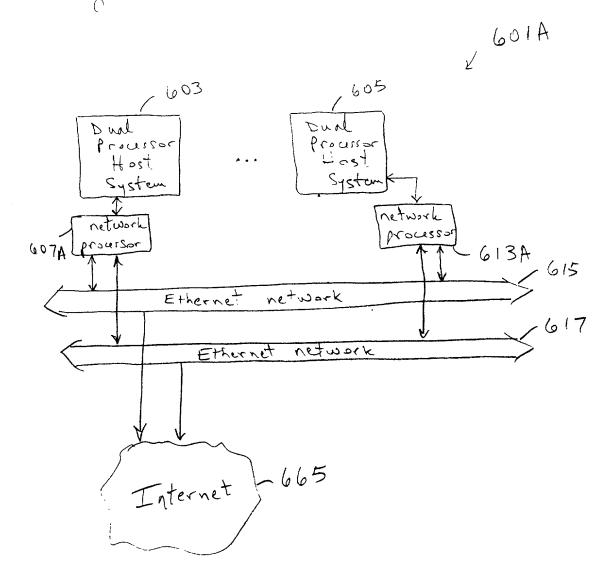


Fig. 170

